2.4 Document Subsets

Some applications require the ability to create a physical representation for an XML document subset (other than the one generated by default, which can be a proper subset of the document if the comments are omitted). Implementations of XML canonicalization that are based on XPath can provide this functionality with little additional overhead by accepting a node-set as input rather than an octet stream. The processing of an element node ***E*** MUST be modified slightly when an XPath node-set is given as input and the element's parent is omitted from the node-set. This is necessary because omitted nodes SHALL not break the inheritance rules of inheritable attributes [C14N-Issues] defined in the xml namespace.

[Definition:] **Simple inheritable attributes** are attributes that have a value that requires at most a simple redeclaration. This redeclaration is done by supplying a new value in the child axis. The redeclaration of a simple inheritable attribute ***A*** contained in one of ***E***'s ancestors is done by supplying a value to an attribute ***Ae*** inside ***E*** with the same name. Simple inheritable attributes are `xml:lang` and `xml:space`.

The method for processing the attribute axis of an element ***E*** in the node-set is hence enhanced. All element nodes along ***E***'s ancestor axis are examined for the nearest occurrences of simple inheritable attributes in the xml namespace, such as `xml:lang` and `xml:space` (whether or not they are in the node-set). From this list of attributes, any simple inheritable attributes that are already in ***E***'s attribute axis (whether or not they are in the node-set) are removed. Then,

lexicographically merge this attribute list with the nodes of ***E***'s attribute axis that are in the node-set. The result of visiting the attribute axis is computed by processing the attribute nodes in this merged attribute list.

The `xml:id` attribute is not a simple inheritable attribute and no processing of these attributes is performed.

The `xml:base` attribute is not a simple inheritable attribute and requires special processing beyond a simple redeclaration. Hence the processing of ***E***'s attribute axis needs to be enhanced further. A "join-URI-References" function is used for `xml:base` fix up. It incorporates xml:base attribute values from omitted xml:base attributes and updates the xml:base attribute value of the element being fixed up, as follows.

An `xml:base` fixup is performed on an element E as follows. Let ***E*** be an element in the node set whose ancestor axis contains successive elements ***En***...***E1*** (in reverse document order) that are omitted and ***E***=***En+1*** is included. (It is important to note that ***En..E1*** is for contiguously omitted elements, for example only `e2` in the example in section 3.8.) The fix-up is only performed if at least one of ***E1*** ... ***En*** had an `xml:base` attribute. In that case let ***X1*** ... ***Xm*** be the values of the `xml:base` attributes on ***E1*** ... ***En+1*** (in document order, from outermost to innermost, ***m*** <= ***n+1***). The sequence of values is reduced in reverse document order to a single value by first combining ***Xm*** with ***Xm-1***, then the result with ***Xm-2***, and so on by calling the "join-URI-References" function until the new value for ***E***'s `xml:base` attribute remains. The result may also be null or empty (`xml:base=""`) in which case `xml:base` MUST NOT be rendered.

Note that this xml:base fixup is only performed if an element with an xml:base attribute is removed. Specifically, it is not performed if the element is present but the attribute is removed.

The join-URI-References function takes an xml:base attribute value from an omitted element and combines it with other contiguously omitted values to create a value for an updated xml:base attribute. A simple method for doing this is similar to that found in sections 5.2.1, 5.2.2 and 5.2.4 of RFC 3986 with the following modifications:

- Perform RFC 3986 section 5.2.1. " Pre-parse the Base URI" modified as follows.
    - The scheme component is not required in the base URI (Base). (i.e. Base.scheme may be null)
    - Replace a trailing ".." segment with "../" segment before processing.

- 5.2.4. "Remove Dot Segments" is modified as follows:
    - Keep leading "../" segments
    - Replace multiple consecutive “/” characters with a single “/” character.
    - Append a “/” character to a trailing “..” segment

- The "Remove Dot Segments" algorithm is modified to ensure that a combination of two xml:base attribute values that include relative path components (i.e., path components that do not begin with a '/' character) results in an attribute value that is a relative path component.

- Perform RFC 3986 section 5.2.2. "Transform References" modified as follows to ignore the fragment

part of R

- After parsing R set R.fragment = null

Then, lexicographically merge this fixed up attribute with the nodes of ***E***'s attribute axis that are in the node-set. The result of visiting the attribute axis is computed by processing the attribute nodes in this merged attribute list.

Attributes in the XML namespace other than `xml:base`, `xml:id`, `xml:lang`, and `xml:space` MUST be processed as ordinary attributes.

The following examples illustrate the modification of the “Remove Dot Segments” algorithm:

- "abc/" and "../" should result in ""
- "../" and "../" are combined as "../../" and the result is "../../"
- ".." and ".." are combined as "../../" and the result is "../../"

To illustrate the last example, when the elements b and c are removed from the following sample XML document, the correct result for the xml:base attribute on element d would be “../../x”:

```
<a xml:base="foo/bar">
  <b xml:base="..">
    <c xml:base="..">
      <d xml:base="x">
      </d>
    </c>
```


3.4 Character Modifications and Character References

Input Document	<!DOCTYPE doc [<!ATTLIST normNames attr NMTOKEN]> <doc> <text>First line
Se <value>2</value> <compute><![CDATA[value>"0" & <compute expr='value>"0" &amp ?"valid":"error"'>valid</compute <norm attr=' ' & <normNames attr=' A &#x20 <normId xml:id=' ' &#x </doc>
Canonical Form	<doc> <text>First line
 Second line</text> <value>2</value> <compute>value>"0" &&a <compute expr="value>"0& ?"valid":"error&q <norm attr=" '
& <normNames attr="A
 <normId xml:id="'
& </doc>

Demonstrates:

- Character reference replacement
- Attribute value delimiters set to quotation marks (double quotes)
- Attribute value normalization
- CDATA section replacement
- Encoding of special characters as character references in attribute values (&, <, ",
,
, 	)

- Encoding of special characters as character references in text (&, <, >,
)

Note: The last element, `normId`, is well-formed but violates a validity constraint for attributes of type ID. For testing canonical XML implementations based on validating processors, remove the line containing this element from the input and canonical form. In general, XML consumers should be discouraged from using this feature of XML.

Note: Whitespace character references other than   are not affected by attribute value normalization [XML].

Note: In the canonical form, the value of the attribute named `attr` in the element `norm` begins with a space, an apostrophe (single quote), then *four* spaces before the first character reference.

Note: The `expr` attribute of the second `compute` element contains no line breaks.

3.7 Document Subsets

Input Document	`<!DOCTYPE doc [`
	`<!ATTLIST e2 xml:space (default|`
	`]>`
	`<doc xmlns="http://www.ietf.org"`
	`  <e1>`
	`  <e2 xmlns="">`
	`  <e3 xml:id="E3"/>`
	`  </e2>`
	`  </e1>`
	`</doc>`

Document Subset Expression	`<!-- Evaluate with declaration x`
	
	`(//. | //@* | //namespace::*)`
	`[`

	`self::ietf:e1 or (parent::iet`
	`or`
	`count(id("E3")|ancestor-or-se`
	`]`

Canonical Form	`<e1 xmlns="http://www.ietf.org"` `xml:id="E3" xml:space="preserve"`

Demonstrates:

- Empty default namespace propagation from omitted parent element
- Propagation of attributes in the `xml` namespace in document subsets
- Persistence of omitted namespace declarations in descendants

Note: In the document subset expression, the subexpression `(//. | //@* | //namespace::*)` selects all nodes in the input document, subjecting each to the predicate expression in square brackets. The expression is true for `e1` and its implicit namespace nodes, and it is true if the element identified by E3 is in the `ancestor-or-self` path of the context node (such that ancestor-or-self stays the same size under union with the element identified by E3).

Note: The canonical form contains no line delimiters.

3.8 Document Subsets and XML Attributes

Input Document	`<!DOCTYPE doc [`
	`<!ATTLIST e2 xml:space (default|p`
	`]>`
	`<doc xmlns="http://www.ietf.org"`
	`xml:base="something/else">`
	`<e1>`
	`<e2 xmlns="" xml:id="abc" x`
	`<e3 xml:id="E3" xml:base`

	</e2> </e1> </doc>
Document Subset Expression	<!-- Evaluate with declaration xm (//. | //@* | //namespace::*) [self::ietf:e1 or (parent::ietf or count(id("E3")|ancestor-or-sel self::node())]
Canonical Form	<e1 xmlns="http://www.ietf.org" x xml:base="something/else"><e3 xml xml:base="something/bar/foo" xml:

Demonstrates:

- `xml:id` not inherited.
- simple inheritable XML attribute inherited (`xml:space`)
- `xml:base` fixup performed

Appendix A

Use the material in the table in Appendix A following "Some Examples" as given in http://lists.w3.org/Archives/Public/public-xml-core-wg/2007Jun/att-0050/Apendix_20060625.html

Add the following text before table:

The following informative table outlines example results of the modified Remove Dot Segments algorithm described in Section 2.4.